Optimizing QoS in Wireless Sensors Networks using a Caching Platform

Rémy Léone[1], Paolo Medagliani[2] and Jérémie Leguay[2]
[1]*UPMC, 4 Place Jussieu, Paris, France*
[2]*Thales Communications & Security, 4 Avenue des Louvresses, Gennevilliers Cedex, France*

Keywords: Wireless Sensor Networks, Contiki, CoAP, Service Oriented Architecture, Embedded Web Services, Network Architecture, Internetworking.

Abstract: This paper addresses monitoring and surveillance applications using Wireless Sensors Networks (WSNs). In this context, several remote clients are interested in receiving the information collected by the nodes of a WSN. As WSN devices are most of the time constrained in energy and processing, we present a caching architecture that will help reducing unnecessary communications and adapting the network to application needs. Our aim here is to cache information in order to improve the overall network lifetime, while meeting requirements of external applications in terms information freshness. We first describe and evaluate the performance of our caching system using a Constrained Application Protocol (CoAP)-HTTP proxy. We then extend this work by showing how the cache could be enriched and exploited using cross-layer data. Based on information from routing packets and estimations of nodes power consumption, we derive an optimization strategy which allows to either maximize the user satisfaction, expressed in terms of freshness of cached data, in the presence of constraints on network lifetime, or jointly maximize network lifetime and user satisfaction, obtaining a set of non-dominated Pareto optimal solutions.

1 INTRODUCTION

Sensors are an interface between a physical phenomenon of interest and a numerical information world (Kovatsch et al., 2012). Recent developments in technology have brought to the use of Wireless Sensor Networks (WSNs) in the *Internet of Things* (IoT), a new vision of the Internet aiming at pushing IP connectivity into smart objects. Nodes inside a WSN are typically low-power and low-processing constrained devices. The networks formed using these devices are commonly referred to as Low-power and Lossy-Networks (LLNs). Due to the typical application fields of a WSN, such as road tunnel fire monitoring, intrusion detection, wildlife monitoring and surveillance scenarios, devices are required to operate for long periods of time, for instance decades, without human intervention, thus the need for high energy efficiency and self configuring capabilities. WSNs are proving to be one of the most interesting innovation in logistic, environment and military technologies (Akyildiz et al., 2002). Due to their constrained nature, it is necessary to introduce optimization and self-configuring strategies to increase their efficiency and autonomy.

First, in this paper we present an architecture for caching data collected from a WSN. When a request from a remote client arrives to the caching node, if it has already a stored value which is “fresh” enough, it replies to the remote client without forwarding the request inside the WSN. This paper presents an implementation and the evaluation of this caching solution within a framework running the Constrained Application Protocol (CoAP) protocol. Relying on CouchDB to store data, our caching system runs on the gateway node of a WSN and it is coupled with Californium, a REST proxy for CoAP. To the best of our knowledge, this is the first paper presenting a working HTTP-CoAP proxy and evaluating the impact of CoAP caching on energy consumption in a WSN.

Then, we extend this work by showing how the caching mechanism could be enriched with cross-layer data and exploited to reach energy optimization. Relying on the routing tree created by the RPL protocol and on a simple analytical model describing the energy consumption of the WSN nodes, we will derive an optimization strategy which allows to either meet a given network lifetime while satisfying Quality of Service (QoS) on the freshness of the cached values or jointly maximize QoS and network lifetime.

The remainder of this paper is structured as follows: In Section 2, we present a list of related works. In Section 3, we introduce the main WSN protocols we will consider in this paper. Then, in Section 4 we describe the caching architecture we have implemented and we show the performance improvements given to the use of the caching. In Section 5, we present the energy model and the cross-layer multi-objective optimization strategy. Finally, Section 6 concludes this paper.

2 RELATED WORK

IETF drafts from CoRE working group are the foundations of the recent innovation in CoAP application layer protocols(Shelby, 2012). Software and architecture used in this paper and in most of academic publications are built on top of these specifications.

Gather information and send it to a single remote user is not enough. In an IoT perspective, this information has to be made easily and immediately available to several clients without any further operation. Web service is a solution to achieve this. Several papers have been published to relate a Service Oriented Architecture (SOA) and WSNs in order to create Web Services. The first paper related to this perspective was (Delicato et al., 2003), where the authors introduced the concept of connection between a WSN and the Internet. In (Leguay et al., 2008), instead the authors presented a SOA architecture adapted to energy constrained devices.

Connecting a WSN with web applications has been explored and several works such as (Kuladinithi et al., 2011; **?**; **?**) introduce an architecture to bridge a WSN to the web. The main point of these works is to present a reliable architecture that implemented a basic HTTP/CoAP (HC) gateway without proxying functionalities.

The use of in-network caching in WSNs in order to avoid redundant transmission is a known technique for efficient time and energy saving. In (Duquennoy et al., 2011), this mechanism is adopted to improve the performance of software updates in incrementally deployed sensor networks

In this paper, leveraging on a simple yet effective HC proxy architecture with caching capabilities, we implement a lifetime optimization strategy, which exploits cross-layering techniques to meet requirements on overall network lifetime, given some constraints on the freshness of cached information.

3 PRELIMINARIES ON WIRELESS SENSOR NETWORKS

Traditional operating systems and Internet protocols are not suited for operating with WSN devices. In order to meet the typical strong requirements of WSNs in terms of energy consumption, it is preferable to let the nodes sleep for an as long as possible, waking them up only when a packet needs to be transmitted. The basic mechanism proposed in Contiki OS, an operating system for low-power devices, is the Contiki MAC (Dunkels, 2011). The nodes remain sleeping, that is with radio interfaces switched off) for the largest part of the duty cycle. When a node needs to transmit a message, it starts cyclically sending the packet as soon as the receiving node acknowledges the reception of the packet. During two consecutive packet transmission attempts, the transmitting node waits for a short time interval in order to detect an eventual incoming acknowledgment packet.

Traditional IPv6 routing protocols are not designed with unstable and intermitting communication links in mind. Therefore, the IETF ROLL Working Group has defined an IPv6 Routing Protocol for Low power and lossy networks (RPL) optimized to deal with multipoint-to-point traffic, as well as point-to-multipoint traffic and point-to-point traffic (Winter et al., 2012). RPL allows the creation of a tree to transfer information to (or from) a root from (or to) the remote nodes. The node which has knowledge of the structure of the tree is the border router.

RESTful web services and the HTTP protocol are widely used to publish the status of resources (Kovatsch et al., 2012). However, web services are not suited for constrained networks due to the exaggerate overhead introduced by HTTP and the presence of the TCP congestion window. For these reasons, the CoRE IETF working group aims at realizing the REST architecture in a suitable form for LLNs through the definition of a protocol referred to as *Constrained Application Protocol* (CoAP) (Shelby, 2012).

When a HTTP client wants to receive the status of a resource running CoAP, the HTTP request must be translated into a CoAP request (and vice versa for the response). Normally, the protocol translation is carried out by a HC proxy. Unlike HTTP, CoAP requests and responses are not sent over a previously established connection. In fact, whilst HTTP is based on the TCP transport protocol, CoAP is based on UDP, which introduces less overhead and has no need for session maintenance. In order to reduce response time and network bandwidth consumption, the CoAP servers or end-points may cache responses.

4 THE CACHING MODEL

In this section we will describe the caching mechanism implementation and we will a simple analytical model which well approximates the performance in terms of requests served by the caching platform.

4.1 Justification

One of the main reasons to develop a caching architecture for WSNs is to let sensors save energy by handling in the caching node the information that has already been requested by a remote client. In this way the WSN devices can continue to stay in a sleep state since remote requests are directly served by the caching node.[1] In addition, exploiting suitable optimization strategies and cross-layering techniques, we can also use the information stored in the cache in order to compute route optimization and graceful degradation on the QoS of the WSN, in order to meet a given overall network lifetime.

According to the CoAP protocol, the goal of the cache is to reuse a previously stored, if recent enough, response to satisfy a client request, reducing energy consumption in the WSN. If the collected value is older than a given threshold parameter, then a request to the resource is performed. Otherwise, the value stored in the cache is directly sent back to the client. According to the protocol drafts and recommendations (Shelby, 2012; **?**), how the proxy actually satisfies the request is an implementation choice. However, we can assume several scenarios where the border router/proxy may be resource constrained. Therefore, in some cases this solution may not be appropriate and one could prefer to avoid the use of such a proxy, falling back to an end-to-end CoAP communication.

4.2 The Caching System Architecture

The caching system has been implemented relying on the IETF protocols. We made this choice both because some libraries, which can be reused to speed up the implementation, are already available and because IETF protocols have been specifically designed to meet the constraints typical of IoT architectures. In order to test and validate the performance of our caching architecture, we have chosen Contiki OS, since it provides for a simulation environment like Cooja and implementations of the RPL and CoAP protocols. As one of the goal of this paper is energy saving inside the WSN, we have decided to use ContikiMac MAC protocol, which provides for duty cycle capabilities and it has been optimized to not occupy too much RAM memory with its footprint. In addition, implementations of CoAP and RPL protocols are already available in Contiki OS, thus allowing us to quickly have a working and running testbed.

[1]We remark that power consumption in sleep state is orders of magnitude lower than those in reception and transmission states.

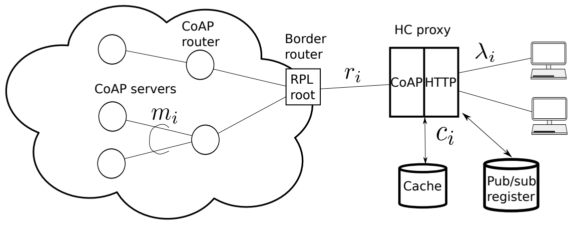


Figure 1: Scheme of the implemented caching system.

In Figure 1, we show a logical scheme of the implemented caching system, where *N* CoAP servers expose one resource to be queried. The running routing protocol is RPL. All the requested information and the notifications from the nodes are gathered to the root of the RPL tree, which overlaps with the WSN gateway. The HC proxy will then send to the final user the requested information.

Our HC proxy implementation is based on the Californium open source framework (Kovatsch et al., 2012). The main reason of this choice is that Californium already implements in JAVA a basic set of CoAP functionalities. We have then introduced two information storing mechanisms. The first one is the caching database, where all the information from the WSN are gathered and stocked and for which we have chosen CouchDB, since it offers a REST HTTP API to query stored results. The second one is a subscriptions register to efficiently manage the subscribers to the CoAP resources.

The arrival rate of each request made by a remote user to the *i*-th resource is distributed as a Poisson process of parameter λ_i (dimension: [s^{-1}]). The requests are intercepted by the HC proxy which handles also the eventual response from the given CoAP server. If the proxy has a stored value which is fresh enough, that is whose lifetime is smaller than a given value c_i (as explained below), it directly replies to the request from a remote client, without forwarding it into the WSN. Otherwise, if the required value is not present or it is older than c_i, it transfers the request to the *i*-th CoAP server. Additionally, the proxy stores the sensor responses in the cache, in order to make them available for other eventual incoming requests.

A similar approach is used for the publish-subscribe register. In the case of observation requests issued by a remote client, the proxy handles them by maintaining a list of observed resources and a list of

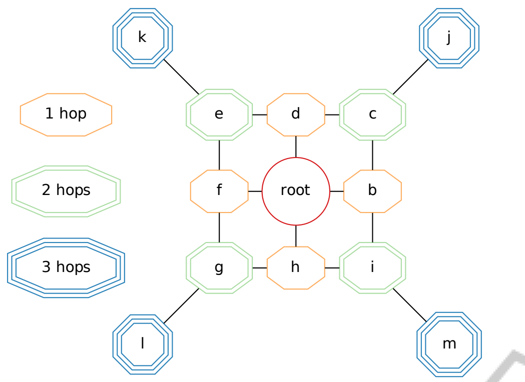


Figure 2: The considered network topology. $N = 12$ nodes (depicted as octagons) are placed in a regular grid with the root (round node) in the middle. The number of lines of the contour of each octagon refers to the distance, in terms of hops, from the root.

interested clients. Each time a notification for a resource update is sent from the node to the proxy, the proxy will forward this messages to all the interested subscribers.

Caching and observe mechanisms can operate together as well. For instance, an information already requested by an observe request can be cached and made available to another request not related to the previous observation. In fact, the caching and the observation mechanism are independent from each other. Nevertheless, the two system should be used together because information obtained by observation can help the caching operations.

4.3 Experimental Validation of the Caching Architecture

4.3.1 Experimental Setup

In order to evaluate the previously described caching model, we introduce the following simulation setup. A grid of $N = 12$ Contiki nodes running CoAP server is deployed. These Contiki-CoAP nodes are emulated using COOJA. Due to limitations of Contiki tools to obtain and analyze the energy consumption, we have preferred using the number of requests served by the HC proxy as a metric to measure of saved energy. In order to have a small memory footprint, each of the CoAP server will only have a single resource and this resource will be a fixed text message. An illustrative example of the considered topology is depicted in Figure 2. According to the scheme shown in Figure 1, a remote client first randomly chooses one of the CoAP servers and then issues to it a `GET` request for knowing the status of the selected resource. The interarrival time between two consecutive requests is distributed according to an exponential distribution of parameter λ.[2] Since the requests for the CoAP resources are equiprobable, exploiting the superposition propriety of the Poisson processes, we can say that the arrival rate of the requests for each node is still a Poisson distribution of parameter $\lambda_i = \lambda/N$, where i denotes the resource on the i-th node, therefore the arrival rate of the requests for a generic i-th node is a Poisson process of parameter λ_i.

The requests from the remote client are intercepted by the HC proxy, which translates HTTP requests into CoAP requests and, conversely, translates the CoAP responses into HTTP responses. In addition, the proxy caches the responses carried out by the CoAP nodes in order to make them available for eventual other incoming request. The freshness of the cached value for the i-th node is denoted as c_i. If a request for the status of the i-th resource arrives, the proxy first verifies in the cache if it already has an available value to be sent back to the client, that is a value whose lifetime is smaller than c_i, otherwise it forwards the request to the WSN node to obtain an updated response. Additionally, the HC proxy stores the collected value in the cache for future requests and forward the collected information to the final client. Like in every caching system, refresh parameters are highly critical. If c_i is too large, information won't be sufficiently renewed and cached values may often be different from the real values. If c_i is too small, then the proxy will be underperforming, wasting energy and not reaching the optimal performance.

The simulations have been carried out generating 50 requests from the remote client to the CoAP servers in the WSN. In order to bound possible statistical fluctuations in the simulation results, the simulation results are obtained by averaging over 10 consecutive runs.

The main parameters of the overall system model are listed in Table 1. The power consumption parameters presented in this table have been taken from the internal data-sheet of a prototype sensor node by Thales. Like other well-known commercial sensor nodes using the CC2420 chipcon (e.g., Crossbow MicaZ, Berkeley Telos (Polastre et al., 2005)), power consumption is higher in the reception mode than in the full-power transmission mode.

In order to simplify the simulations, we have considered the same values of λ_i, c_i, and the same duty cycle for each CoAP node in the WSN.

[2]We remark that is λ is the arrival rate of the requests to the HC proxy.

Table 1: System parameters considered in the simulations.

Node transmission rate	R	250 kbps
Interval between two consecutive strobed preambles	T_{p}	0.4 ms
Time to detect an incoming Ack packet	T_{d}	0.16 ms
`GET` packet length	L_{GET}	87 bytes
Response packet length	L_{Ans}	96 bytes
Ack transmission duration	S_{Ack}	0.608 ms
Transmission power consumption	P_{Tx}	0.0511 W
Reception power consumption	P_{Rx}	0.0588 W
Sleep power consumption	P_{sleep}	$2.4 \cdot 10^{-7}$ W
Number of nodes in the network	N	12

4.3.2 Analytical Model of the Caching Architecture

The mechanism of caching can be summarized as follows: a client issue a request to one of the nodes in the network. This means that if the proxy doesn't have an updated replica of the required value, it transfers the request to the WSN, caching the response for eventual other requests to the same resource. The interarrival time T_i (dimension: [s]) between two consecutive requests can be modeled as an exponentially distributed random variable of parameter λ_i. Since in our simulation the destination of the request is randomly chosen, on average, we can say that we must wait for N consecutive requests before having a new request on the same node i. Therefore, assuming that λ_i is the same for all the N nodes in the network, the time interval between the different requests to the same node is distributed with an exponential law of parameter $T = NT_i = N/\lambda_i$.

The cache miss ratio (MR) can be defined as the probability that a request is not served by the caching architecture. Therefore, MR corresponds to the probability that the interarrival time between two consecutive requests to the same node is larger than c_i, that is

$$MR = \int_{c_i}^{\infty} \frac{e^{-\frac{t}{T}}}{T} \mathrm{d}t = e^{-\frac{c_i}{T}}. \tag{1}$$

Therefore, the cache hit ratio (CH) can be expressed as the complementary of MR, that is

$$CH = 1 - e^{-\frac{c_i}{T}}. \tag{2}$$

According to the scheme depicted in Figure 1, the remote clients send a resource observation request to one of the N randomly chosen CoAP servers. The request arrival rate is distributed according to a Poisson distribution of parameter λ_i, where i denotes the node to which the request is directed. As explained in the previous subsection, for every observation it is possible to define a freshness parameter c_i, that is for how long the collected data is valid before the data is discarded by the proxy and a new data request must be issued to the CoAP server.[3] In truth, as explained later, for every observed value we can define a minimum and a maximum admittable freshness values, denoted as $c_{i_{\min}}$ and $c_{i_{\max}}$ respectively.

If the resource required by the remote client is not in the cache or the cached value is outdated, the proxy issues a CoAP request to the given CoAP server. Given that a request for the i-th node arrives, on average, every T_i s, it can be proved that the average period r_i (dimension: [s]) between two consecutive requests to the same node can be defined as:

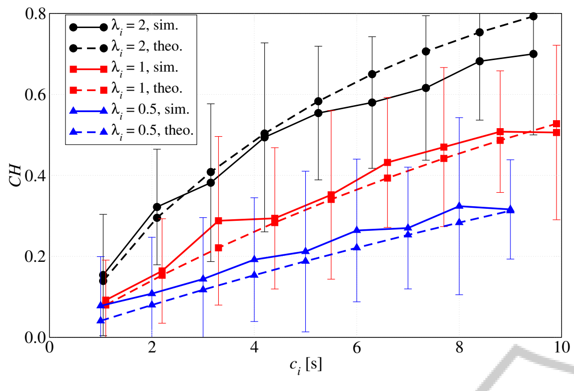
$$r_i = \begin{cases} \lceil \frac{c_i}{T_i} \rceil T_i & \text{if } T_i \le c_i \\ T_i & \text{else.} \end{cases} \tag{3}$$

In the following subsection, we will provide the performance of the caching system in terms of cache hit ratio and network lifetime.

4.3.3 Performance Validation of the Caching Architecture

In order to evaluate the performance of the implemented caching system, we first evaluate the cache hit ratio CH as a function of the freshness parameter c_i. We indicate, for each performance curve, the confidence interval 2σ, where σ is the standard deviation, over consecutive simulation runs, with respect to their average value. In Figure 3 (a), the results are depicted. Both simulation (solid lines) and theoretical (dashed lines) results ares shown considering different values of the parameters λ_i. Of course, the higher the value of λ_i, the higher the number of requests per unit of time. All the curves shown in Figure 3 (a) are growing when the cache duration parameter increases. Clearly, the larger the cache duration, the

[3]In our derivation, we assume that each node has only one available CoAP resource, therefore the freshness of the i-th observed value corresponds to the freshness of the i-th node.



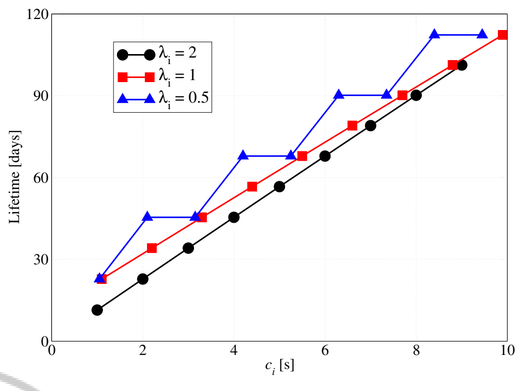


Figure 3: (a) *CH* as a function of the cache freshness c_i. Simulation (solid lines) and theoretical (dashed lines) results are presented. (b) Network lifetime as a function of the cache freshness c_i. In both cases, different values of λ_i have been considered: $\lambda_i = 2$ (lines with circles), $\lambda_i = 1$ (lines with squares), and $\lambda_i = 0.5$ (lines with triangles).

higher the probability that the request is served by the proxy, not being forwarded to the WSN nodes. Similarly, for a given value of cache duration, the higher the number of arrival per unit of time, the higher the probability that the proxy replies to the client with a cached value. The beneficial effect of the use of caching is not limited only to the energy saving related to the lower number of packets transferred to the WSN with the use of proxy. When the traffic is high, in fact, it becomes more and more likely that a cached value may serve several requests from some remote clients. Therefore, the introduction of a caching system allows to reduce the requests transferred to the WSN and, consequently to significantly reduce packet retransmissions or losses.

The use of a caching system also allows to save energy and extend the network lifetime, defined as the time required to the network to have at least one node running out of energy. In Figure 3 (b) we can see the impact of caching on the lifetime of our system. As the intuition may suggest, the larger c_i, the larger the network lifetime, since a larger number of requests are served by the cache without being transferred to the WSN. The “stair” behavior of the curve with $\lambda_i = 0.5$ is due to the fact that, according to the definition of r_i in (3), different values of c_i lead to the same number of requests transferred to the WSN.

5 ADVANCED CACHE EXPLOITATION

5.1 Parameters of the Optimization Framework

The choice of the optimal caching lifetime is application dependent. According to the energy level of a node, we can decide to increase the lifetime of the cached values, accepting less updated values but, on the other side, extending the network lifetime, due to the reduced number of requests transferred to the WSN.

In order to derive an optimization framework, we must rely on the hypothesis that the WSN nodes are static and the quality of the communication links are fair enough to consider the transmissions as error free, so no energy is wasted for packet retransmissions. We also assume that no transmission phase locking mechanism has been used at the MAC layer. We need to introduce some parameters to describe the optimization framework within the caching architecture:

- $c_{i_{\min}}$ and $c_{i_{\max}}$ are the lower and upper bounds, respectively, of the amount of time that a requested information can live in the cache. It’s useless to request an information that is fresher than $c_{i_{\min}}$. Similarly, an information older than $c_{i_{\max}}$ is not meaningful anymore.
- c_i is the cache lifetime of the i-th resource, with $c_i \in [c_{i_{\min}}; c_{i_{\max}}]$. This information can be changed according to the current state of our system in order to fulfill our optimization goals. When an updated information is needed, the c_i parameter can be set to 0, so that the caching node becomes transparent.

It is necessary to define some metrics of interest in order to characterize the optimization strategy introduced in this paper. First of all, we define the network lifetime as the time interval between the network start-up and the first node running out of energy. In our approach, we consider, as a metric of user satisfaction, the freshness c_i of the values stored in the cache, that is, the shorter the cache duration, the higher the user satisfaction. Of course, referring to the results presented in Figure 3 (a), when the duration of the cached values is small, it is likely that an incoming request is transferred to the WSN, thus resulting in energy consumption and reduction of the network lifetime. On the other side, the maximization of the network lifetime requires to minimize the transmissions inside the WSN, that is to keep the duration of the cached values as elevated as possible. The user satisfaction for the *i*-th resource can be then defined as

$$\gamma_i = \frac{c_{i_{\max}} - c_i}{c_{i_{\max}} - c_{i_{\min}}} * 100, \tag{4}$$

that is a user satisfaction of 0% when $c_i = c_{i_{\max}}$ and a user satisfaction of 100% when $c_i = c_{i_{\min}}$.

5.2 Energy Model of the Cached WSN

In order to describe the energy consumption associated to the collection of resources, we assume that the impact of RPL messages is neglectable. Roughly, we can say that the energy consumption of each node is given by the sum of the energy consumptions of its hardware components. For the sake of simplicity, we only integrate in the energy model contributions from the communication sub-unit (radio transceiver). The network lifetime is defined as the time needed for the average residual energy E_r to be lower than a (given) threshold value E_th, which can be used to model the physical behavior of a node. The residual energy of node *i* at time *t*, denoted as $E_{\mathrm{r}_i}(t)$, can be expressed as

$$E_{\mathrm{r}_i}(t) = E_{0_i} - \Omega_{\mathrm{tot}_i} t \tag{5}$$

where E_{0_i} is the initial energy of the *i*-th node and Ω_{tot_i} (dimension: [W]) is the power consumption associated to its communication operations. For the sake of simplicity, we assume that all nodes have the same initial energy. According to the descriptions of the Contiki MAC protocol in Section 3 and in (Dunkels, 2011), there are four possible states for a node: (i) transmission, (ii) reception, (iii) sleep, and (iv) channel listening, with corresponding power consumptions denoted as Ω_{T_i}, Ω_{R_i}, Ω_{S_i}, and Ω_{CL_i} (dimension:[W]), respectively. Before starting the derivation we need to define the period T_{MAC_i} as the time interval between two subsequent active phases of the *i*-th node, and T_{sleep_i} as the duration of the sleep phase of the *i*-th node.[4] The Ω_{tot_i} in (5) can be expressed as follows:

$$\Omega_{\mathrm{tot}_i} = \Omega_{\mathrm{S}_i} + \Omega_{\mathrm{CL}_i} + \Omega_{\mathrm{T}_i} + \Omega_{\mathrm{R}_i} \tag{6}$$

[4]The duration of the active phase T_{act_i} can be evaluated as $T_{\mathrm{act}_i} = T_{\mathrm{MAC}_i} - T_{\mathrm{sleep}_i}$.

where: Ω_{S_i} is the power consumption associated with the sleep phase of the *i*-th node; Ω_{CL_i} is the power required when performing the channel listening operations (over a period of duration T_MAC); Ω_{R_i} is the power used by a node to receive a packet; Ω_{T_i} is the power used to transmit an alert packet.

According to the RPL protocol, there are three kinds of nodes: (i) the CoAP servers, (ii) the routers, acting also as CoAP servers, and (iii) the root of the tree. Since the length of the GET messages and the observed values are different in length, we will distinguish between the time interval required to transmit/receive a GET packet, defined as $S_\mathrm{GET} = L_\mathrm{GET}/R$, and the time required to transmit/receive the response from a CoAP server, defined as $S_\mathrm{Ans} = L_\mathrm{Ans}/R$, where L_GET and L_Ans are the packet lengths of the GET packet and the observation packet, respectively, and R is the transmission rate of the nodes. According to the Contiki MAC protocol, as described in (Dunkels, 2011) when a node needs to transmit a packet, it must be for a period of time $S_{\mathrm{pck-Tx}_\mathrm{Tx}}$ in transmission and for a period of time $S_{\mathrm{pck-Tx}_\mathrm{Rx}}$ in reception, in order to receive the acknowledgment (ACK) message from the receiving node. This period of time spent in transmission of the packet can be expressed as

$$S_{\mathrm{pck-Tx}_\mathrm{Tx}} = \frac{3 + \lfloor \frac{T_{\mathrm{sleep}_i} - S_\mathrm{pck}}{S_\mathrm{pck}} \rfloor}{2} S_\mathrm{pck} \tag{7}$$

whereas the period of time spent in reception during the transmission phase can be expressed as

$$S_{\mathrm{pck-Tx}_\mathrm{Rx}} = \frac{3 + \lfloor \frac{T_{\mathrm{sleep}_i} - S_\mathrm{pck}}{S_\mathrm{pck}} \rfloor}{2} T_\mathrm{d} + S_\mathrm{ack} \tag{8}$$

where S_pck indicates a generic transmitted packet (it must be replaced with S_GET or S_Ans depending on whether the node is transmitting a GET or a response packet), T_d is the time required to successfully detect an acknowledgment from the receiver, and S_ack is the time required to transmit an ACK. Equation (7) has been derived averaging between the best case of packet transmission, that is the node starts transmitting when the receiving node is waken up, and the worst case, that is the receiving node has just switched into the sleep phase when the transmitting node begins sending the packet. Similarly, when a node needs to receive a packet, it spends a part of the time in reception and a part of the time in transmission since it has to send the ACK to the transmitting node. The period of time spent receiving the packet can be expressed as

$$S_{\mathrm{pck-Rx}_\mathrm{Rx}} = \frac{3S_\mathrm{pck}}{2} + T_\mathrm{p} \tag{9}$$

whereas the period of time spent in transmission during the reception phase can be expressed as

$$S_{\text{pck}-\text{Rx}_{\text{Tx}}} = S_{\text{ACK}} \tag{10}$$

where T_{p} is the interval between each packet transmission and S_{ACK} is the duration of the transmission of an ACK message. We point out that equation (9) has been obtained averaging the energy consumption between the best, that is the receiving node needs to receive only the packet once, and the worst case of packet arrival, that is the receiving node wakes up just after the beginning of the transmission of the packet by the transmitting node, so it has to wait for the next packet transmission to correctly receive it.

According to the different nodes allowed by the CoAP protocol, the terms Ω_{T_i} and Ω_{R_i} will be different for a CoAP server, a router, and the root of the tree. In the first case, the terms will be replaced by $\Omega_{\text{CoAP}-\text{T}_i}$ and $\Omega_{\text{CoAP}-\text{R}_i}$, in the second case the terms will be replaced by $\Omega_{\text{Router}-\text{T}_i}$ and $\Omega_{\text{Router}-\text{R}_i}$, whereas in the third the terms will be replaced by $\Omega_{\text{root}-\text{T}_i}$ and $\Omega_{\text{root}-\text{R}_i}$. Considering a generic CoAP server without routing functionalities, which only receives a GET message and transmits the observed value, the power consumption during the transmission phase can be expressed as

$$\Omega_{\text{CoAP}-\text{T}_i} = \frac{P_{\text{Tx}} S_{\text{Ans}-\text{Tx}_{\text{Tx}}} + P_{\text{Rx}} S_{\text{Ans}-\text{Tx}_{\text{Rx}}}}{r_i} \tag{11}$$

whereas the power consumption during the reception phase can be expressed as

$$\Omega_{\text{CoAP}-\text{R}_i} = \frac{P_{\text{Rx}} S_{\text{GET}-\text{Rx}_{\text{Rx}}} + P_{\text{Tx}} S_{\text{GET}-\text{Rx}_{\text{Tx}}}}{r_i} \tag{12}$$

replacing the packet lengths of the GET message and of its related response into equations (7), (8), (9), and (10). The terms P_{Tx} and P_{Rx} denotes the power consumptions of a node in transmission and reception phase. Considering the root node of the network, instead, the power consumptions to transmit a GET message to a generic node i and receive back its response can be expressed as

$$\Omega_{\text{root}-\text{T}_i} = \frac{P_{\text{Tx}} S_{\text{GET}-\text{Tx}_{\text{Tx}}} + P_{\text{Rx}} S_{\text{GET}-\text{Tx}_{\text{Rx}}}}{r_i}$$

and

$$\Omega_{\text{root}-\text{R}_i} = \frac{P_{\text{Rx}} S_{\text{Ans}-\text{Rx}_{\text{Rx}}} + P_{\text{Tx}} S_{\text{Ans}-\text{Rx}_{\text{Tx}}}}{r_i},$$

respectively. Since the root transmits to each of the N associated children, the power consumption to transmit to all the nodes and to receive a packet from all the nodes can be expressed as

$$\Omega_{\text{root}-\text{T}} = \sum_{i=1}^{N} \Omega_{\text{root}-\text{T}_i} \tag{13}$$

and

$$\Omega_{\text{root}-\text{R}} = \sum_{i=1}^{N} \Omega_{\text{root}-\text{R}_i}, \tag{14}$$

respectively. Considering the intermediate routing nodes, since they are in charge of transferring both the request from the root to the CoAP servers and the responses from the CoAP servers to the root, we can say that they act both as root for the children nodes and as CoAP servers for the root. Therefore, the power consumptions to forward the incoming packets can be expressed as

$$\Omega_{\text{Router}-\text{T}_i} = \sum_{j\in m_i} \left(\Omega_{\text{CoAP}-\text{T}_j} + \Omega_{\text{root}-\text{T}_j}\right) + \Omega_{\text{CoAP}-\text{T}_i} \tag{15}$$

and

$$\Omega_{\text{Router}-\text{R}_i} = \sum_{j\in m_i} \left(\Omega_{\text{CoAP}-\text{R}_j} + \Omega_{\text{root}-\text{R}_j}\right) + \Omega_{\text{CoAP}-\text{R}_i}, \tag{16}$$

where m_i, as shown in Figure 1, denotes the number of children associated to the i-th node. The terms at the right-hand side of (15) and (16) have been introduced since these nodes, besides acting as routers, may also behave as CoAP servers.

Finally, the power consumption in the channel listening state can be expressed as

$$\Omega_{\text{CL}_i} = \frac{T_{\text{act}} P_{\text{Rx}}}{T_{\text{MAC}}}, \tag{17}$$

whereas the power consumption in the sleep phase can be denoted as

$$\Omega_{\text{S}_i} = \frac{T_{\text{sleep}_i} P_{\text{Sleep}}}{T_{\text{MAC}}} - \Gamma_{\text{Tx}_i} - \Gamma_{\text{Rx}_i}, \tag{18}$$

where P_{Sleep} is the power consumed in the sleep state, and Γ_{Tx_i} and Γ_{Rx_i} are two corrective terms, described in more detail below. During normal operations the node either performs channel listening and sleep operations or transmits/receives a packet. Γ_{Tx_i} and Γ_{Rx_i} are used to refine the power consumption due to sleep operations. In fact, the sleep and also the transmission and reception intervals overlap for short intervals, so that without these two terms the power consumption budget would be higher than the correct one. In particular, Γ_{Tx_i} can be expressed as

$$\Gamma_{\text{Tx}_i} = \Omega_{\text{S}_i} \frac{\left(\frac{3 + \left\lfloor \frac{T_{\text{sleep}_i} - S_{\text{pck}}}{S_{\text{pck}}} \right\rfloor}{2} \left(S_{\text{pck}} + T_{\text{d}}\right) + S_{\text{ack}} \right)}{T_{\text{MAC}_i}} \tag{19}$$

whereas Γ_{Rx_i} can be expressed as

$$\Gamma_{\text{Rx}_i} = \Omega_{\text{S}_i} \frac{\left(\frac{3 S_{\text{pck}}}{2} + T_{\text{p}} + S_{\text{ack}} \right)}{T_{\text{MAC}_i}} \tag{20}$$

The term Γ_{Tx_i} takes into account the fact that, during transmission operations, such as (i) strobed transmission of the packet over an interval of duration T_{sleep_i}, (ii) packet transmission, and (iii) acknowledgment reception, a node would normally be in the sleep state. Thus, the correction factor Γ_{Tx_i} is necessary, since otherwise the energy consumed by the node with this model would be higher than the real value because reception and transmission operations would overlap with normal sleep operations for a period. Similar considerations can be carried out for the term Γ_{Rx_i}. In fact, when a node is waiting for the acknowledgment window to transmit an ACK message, to receive the preamble, and to receive a packet, it would normally be in the sleep state.

Replacing expressions (19) and (20) into (18) and expressions (11), (12) (if a CoAP node, otherwise (15) and (16) for a router or (13) and (14) for the root), (17), and (18) into (6), it is possible to derive an expression for the energy consumption depending on topology and communication parameters.

5.3 An Optimization Framework

The above introduced analytical model can be used during network configuration in order to find the proper values of c_i which allow to meet a given requirement on minimum network lifetime. Of course, as introduced in Subsection 5.1, considering a minimum and a maximum value of cache duration, the best network lifetime can be reached using $c_i = c_{i_{\max}}$. On the other side, if we want to maximize the average user satisfaction γ, defined as $\gamma = \sum_{i=1}^{N} \gamma_i / N$, we must set $c_i = c_{i_{\min}}$, with the opposite undesirable effect that the lifetime is minimized.

The choice of the suitable values of c_i introduces then a trade-off between network lifetime and user satisfaction. In addition, since some nodes act as routers, decreasing the number of transmissions to the i-th node may affect not only the lifetime of node i, but also the lifetime of the routers which are on the path between node i and the root. In order to solve these problems, we use the Non-dominated Sorting Genetic Algorithm II (NSGA II) (Deb et al., 2002). Given the minimum target lifetime, the interval $[c_{i_{\min}}; c_{i_{\max}}]$, and the network topology generated by the RPL protocol, the solver provides for the best set of c_i parameters which satisfies the constraints on lifetime while maximizing the user satisfaction γ.

Considering a scenario with $c_{i_{\min}} = 1$ s and $c_{i_{\max}} = 9$ s, $\lambda_i = 1$ s^{-1}, and the topology with $N = 12$ nodes shown in Figure 2, we have set some constraints on network lifetime, namely at least 25, 50, and 75 days, and we have evaluated the set of $c_{i_{\text{opt}}}$ which maximizes the user satisfaction while fulfilling the network lifetime requirement. As a comparison, we also show the results obtained without the optimization tool, that is those obtained with $c_i = c_{i_{\min}}$ and $c_i = c_{i_{\max}}$. The results are shown in Table 2. The set of c_i obtained through the optimization tool allows to effectively meet the requirement on network lifetime while maximizing the user satisfaction.

Table 2: Performance results of the optimization strategy compared with the cases with $c_i = c_{i_{\min}}$ and $c_i = c_{i_{\max}}$. For estimating $c_{i_{\text{opt}}}$, target lifetimes of 25, 50, and 75 days have been considered.

c_i	lifetime [days]	γ
$c_{i_{\min}}$	11.4219	100%
$c_{i_{\text{opt}}}^{25}$	25	95%
$c_{i_{\text{opt}}}^{50}$	50	68%
$c_{i_{\text{opt}}}^{75}$	75	49%
$c_{i_{\max}}$	101.256	0%

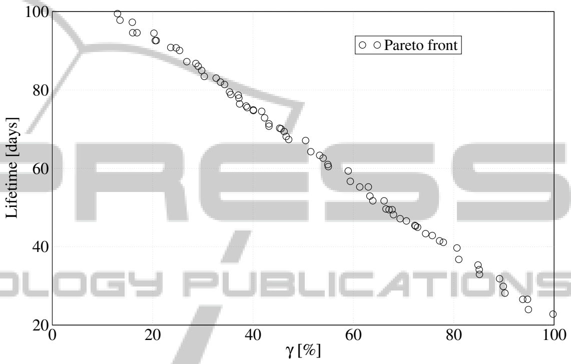


Figure 4: Pareto front of the considered caching architecture.

If no constraints on lifetime are imposed, finding a suitable working point (i.e., a suitable set of values of c_i) for the system can be seen as a multi-objective optimization problem. In this case, the NSGA II algorithm allows to find the non-dominated Pareto front of solutions, that is the set of values which are Pareto optimal. According to the definition of Pareto optimality, a solution is Pareto optimal when it is not possible to improve one objective without reducing at least one of the other objectives. In Figure 4, the Pareto front for the presented caching architecture is shown. According to the presented results, it is possible to find out a set of suitable working points which allows to achieve the best possible network configuration. The solver associates to each of these points a set of parameters c_i which can be used at the network start-up to properly tune the cache duration.

We point out that the above presented optimization strategy has been evaluated according to a specific definition of QoS. However, it can be easily generalized in order to encompass other objective functions or other constraints in the optimization process.

6 CONCLUSIONS

This paper has addressed the problem of energy efficient QoS optimization in WSNs using cross-layering techniques and exploiting a specifically introduced caching platform. We have first presented the implementation of a caching solution based on a proxy node which is in charge of answering, if a cached value is available, to a request coming from a remote client without transferring it to the WSN. Simulation results show that the introduction of a caching architecture has an impact in terms of energy saving on the system performance, since it allows to reduce the transmissions inside the WSN. Then, we have introduced an optimization framework which, exploiting the information collected by the RPL protocol and given a set of constraints on the minimum and maximum values of cache duration, allows to optimally configure the values of the caching lifetimes. The proposed optimization strategy allows to either find suitable solutions in the presence of constraints on network lifetime or to find out the optimal non-dominated set of solutions in the case of multi-objective optimization.

Further works include the real-time change of the routing paths, in order to save the nodes which are running out of energy, and large-scale experiments, using the Senslab platform (Senslab Website, 2008), in order to get real energy consumption data from physical nodes.

ACKNOWLEDGEMENTS

This work is funded by the European Community's Seventh Framework Programme, area "Internetconnected Objects", under Grant no. 288879, CALIPSO project - Connect All IP-based Smart Objects. The work reflects only the authors' views; the European Community is not liable for any use that may be made of the information contained herein.

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197.

Delicato, F., Pires, P., Pinnez, L., Fernando, L., and da Costa, L. (2003). A flexible web service based architecture for wireless sensor networks. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 730–735. IEEE.

Dunkels, A. (2011). The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science.

Duquennoy, S., Wirström, N., Tsiftes, N., and Dunkels, A. (2011). Leveraging IP for Sensor Network Deployment. In *Proc. of the Work. on Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*.

Kovatsch, M., Mayer, S., and Ostermaier, B. (2012). Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things. In *Proc of the 6th Int Conf on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), Palermo, Italy*.

Kuladinithi, K., Bergmann, O., Pötsch, T., Becker, M., and Görg, C. (2011). Implementation of CoAP and its Application in Transport Logistics. *Proc. IP+ SN, Chicago, IL, USA*.

Leguay, J., Lopez-Ramos, M., Jean-Marie, K., and Conan, V. (2008). An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. In *Local Computer Networks. LCN 2008. 33rd IEEE Conf*, pages 740–747. IEEE.

Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: enabling ultra-low power wireless research. In *Proc. of the 4th Int. Symp. on Information Processing in Sensor Networks (*IPSN 05), pages 364 – 369, Piscataway, NJ.

Senslab Website (2008). http://www.senslab.info/.

Shelby, Z. (2012). Constrained Application Protocol (CoAP). Internet-Draft draft-ietf-core-coap-09, Internet Engineering Task Force. Work in progress.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard).